

Chapter 6

CMOS Functional Cells

In the previous chapter we discussed methods of designing layout of logic gates and building blocks like transmission gates, multiplexers and tri-state inverters.

In this chapter we consider design methodology, structures and layouts of functional cells like 1-bit adders, latches and flip-flops.

We start with a design methodology based on application of circuit graphs and layout matrices.

6.1 Design methodology

6.1.1 Problem formulation

The problem that we would like to study can be stated as follows.

Given a schematic of a CMOS cell, create its mask layout in a form of a **linear structure** of (n-MOS, p-MOS) pairs placed in two parallel rows, each in its respective well.

In other words, the layout of a functional cell should be based on a pseudo-serial connection of nMOS transistors in a p-well and pMOS transistors in the n-well. This will form continuous horizontal rows of nMOS and pMOS transistors.

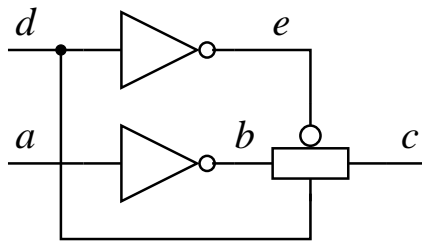
If such a continuous connection is electrically not possible, discontinuities in diffusion (active regions) must be created.

Complementary transistors should be aligned vertically and their gates should be driven from the vertical polysilicon lines.

We will aim at **minimisation of discontinuities** in the diffusions areas, which indirectly reduces the number of space consuming contacts between layers.

6.1.2 An illustrative example.

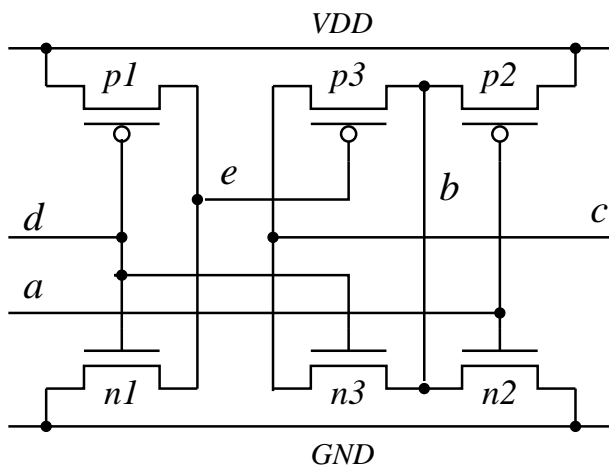
As an illustrative example, we consider a **generic tri-state inverter** presented earlier in sec.5.10. A logic diagram and a schematic of such a tri-state inverter with an additional inverter to generate a complementary control signal e is presented in Figure 6.1.



$$c = \begin{cases} \bar{a} & \text{if } d = 1 \\ Z & \text{if } d = 0 \end{cases}$$

a. Logic diagram of a conceptual tri-state inverter

A tri-state inverter inverts the input signal a when the enable signal d is high, otherwise it disconnects the output c .



b. A "first-guess" schematic (one discontinuity)

Figure 6.1: A logic diagram and a schematic of a tristate inverter based on the transmission gate.

The cell consists of 3 pairs of transistors ($p1, n1; p2, n2; p3, n3$) and seven electrical nodes (V, a, b, c, d, e, G , where V and G denote V_{DD} and GND , respectively). For our purposes, five nodes, V, b, c, e, G , which are connected to sources or drains of the MOS transistors, are of prime interest.

The structure of the circuit can be described by the following **layout matrix**.

$$\mathcal{L}_1 = \begin{bmatrix} V & (p1) & e!c & (p3) & b & (p2) & V \\ & d & & e & & a & \\ & d & & d & & a & \\ G & (n1) & e!c & (n3) & b & (n2) & G \end{bmatrix} \quad (6.1)$$

We note two discontinuities in the diffusion areas between nodes e and c which are marked with '!'.

To improve the layout of the tristate inverter of Figure 6.1, the layout matrix can be modified in such a way that the nodes e and c are moved out to the boundaries of the active area, that is, to the first and the last columns of the layout matrix, which eliminates the discontinuity. The resulting layout matrix, schematic and the equivalent layout of the tristate inverter are as follows:

$$\mathcal{L}_2 = \begin{bmatrix} e & (p1) & V & (p2) & b & (p3) & c \\ & d & & a & & e & \\ & d & & a & & d & \\ e & (n1) & G & (n2) & b & (n3) & c \end{bmatrix} \quad (6.2)$$

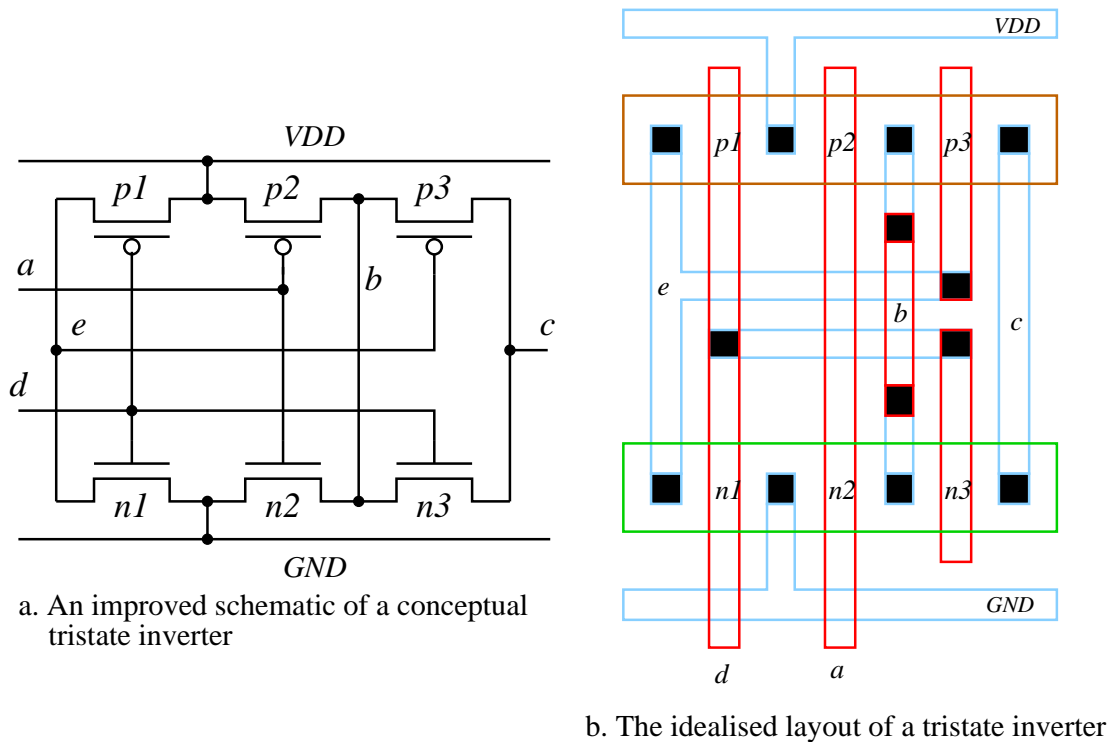


Figure 6.2: An improved schematic and an idealised layout of a generic tristate inverter

In the idealised layout of Figure 6.2 coloured boxes represent respective material, small black squares representing contact cuts.

The structure of the **tristate inverter** cell can be further **improved** by elimination of the node *b*, as discussed in sec.5.10.

The symbol, schematic and the layout of the modified tristate inverter is given in Figure 6.3.

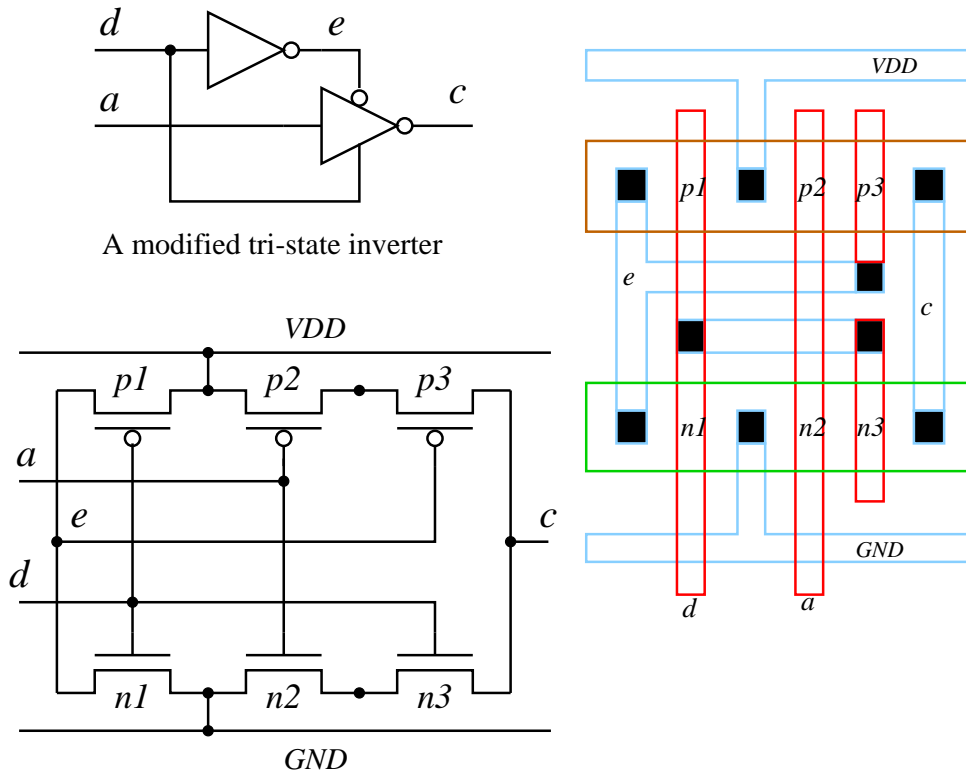


Figure 6.3: The symbol, schematic and the layout of the modified tri-state inverter

The layout can be described by the following layout matrix:

$$\mathcal{L}_3 = \begin{bmatrix} e & (p1) & V & (p2) & (p3) & c \\ & d & & a & e & \\ & d & & a & d & \\ e & (n1) & G & (n2) & (n3) & c \end{bmatrix} \quad (6.3)$$

In a more formal way, we can state that the reason that the simplification of the layout of Figure 6.1 (eqn 6.1) into a form as in Figure 6.2 (eqn 6.2) has been possible is clearly visible from the **circuit graphs** of the two diffusion areas, which are presented in Figure 6.4.

The graphs of such a simple, linear structure, are known as **semi-Eulerian graphs**.

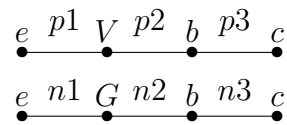


Figure 6.4: Circuit graphs representing two diffusion areas for the tristate inverter

Hence, if circuit graphs are semi-Eulerian, equivalent diffusion paths have no discontinuities.

6.1.3 Basic facts from the graph theory

We will need the following the basic definitions and theorems used in the graphs theory.

- A **connected graph** is a graph in which any two **nodes** (vertices) are connected by a path.
- A **path** is a **walk** in which no edge appears more than once.
- A **walk** is a sequence of edges.
- A **trail** is a walk in which all the edges are **distinct**.
- A connected graph is called **Eulerian** if there exists a closed trail which includes every edge of the graph; such a trail is called an **Eulerian trail**.
- If the trail is not closed, as is the case in the layout problem, we call it a **semi-Eulerian** trail.
- The **degree** of a node (vertex) of a graph is the number of edges incident at that node.

The fundamental for the layout problem theorem states that:

A connected graph is semi-Eulerian if and only if there are 0 or 2 nodes of odd degree

If a semi-Eulerian graph has exactly **two nodes of odd degree**, then any semi-Eulerian trail must **start at one of them and terminate at the other**.

Note that a graph cannot have exactly one node of odd degree (the handshaking lemma).

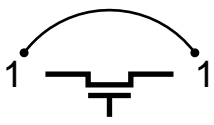
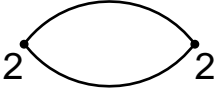
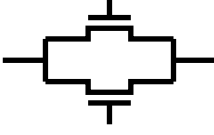
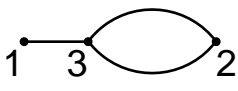

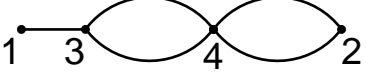
	Number of nodes of the odd degree :	Does a semi- Eulerian trail exist?
	2	yes
	0	yes
	0	yes
	2	yes
	4	no
	2	yes

Figure 6.5: Simple graphs annotated with degrees of nodes

In Figure 6.5 a collection of simple graphs is given. Each node is annotated with its degree.

6.1.4 Layout Generation Methodology

- Form the graph of the circuit and calculate the degree of each node.
- If zero or two nodes are of odd degree, semi-Eulerian trails for n-diffusion and p-diffusion exist. Find these trails by traversing all the edges representing n-MOS and p-MOS transistors in the circuit graph, respectively. A constructive method of finding Eulerian trails is known as Fleury's algorithm.
- In order to simplify the layout further align vertically the corresponding p-MOS and n-MOS transistors.

If Eulerian trails do not exist:

- It may be possible to modify the circuit knowing its logical and functional properties,
- Alternatively, duplicate some transistors, thus increasing the degree of corresponding nodes,
- If discontinuities must exist, it may be possible to minimise their number.

Partial solutions to the above problems will be discussed in the next section.

6.1.5 Designing a CMOS functional cell – an illustrative example.

In this section we will employ the layout generation methodology to the typical CMOS functional cells, in order to demonstrate its application in practical situations.

Firstly, we concentrate on the problem of arranging p-diffusion and n-diffusion areas in an optimal way, that is forming the Eulerian trails, if possible.

Related to this issue is the optimal way of arranging first metal paths, in the context of interconnections between sources and drains of transistors.

Occasionally, we refer to the optimal way of arranging polysilicon paths in the context of the gate signals. The general optimality problem of the complete circuit layout is not addressed.

A static CMOS functional cell consists of two complementary circuits often referred to as a pull-up circuit (p-MOS transistors) and a pull-down circuit (n-MOS transistors), respectively.

The p-MOS and n-MOS transistors always appear in pairs, and such a pair is often driven by the same gate signals. Therefore, in the layout, transistors belonging to the pair are kept aligned vertically, which simplifies the layout further.

Obtaining Eulerian trails by circuit manipulations

Consider a circuit diagram of a combinational cell implementing the function:

$$e = \overline{a + b \cdot c + d}$$

p-MOS circuit: $e_p = \bar{a} (\bar{b} + \bar{c}) \bar{d}$
n-MOS circuit: $e_n = a + b \cdot c + d$

A schematic and equivalent layout graphs for p-type and n-type diffusion for the above cell is given in Figure 6.6.

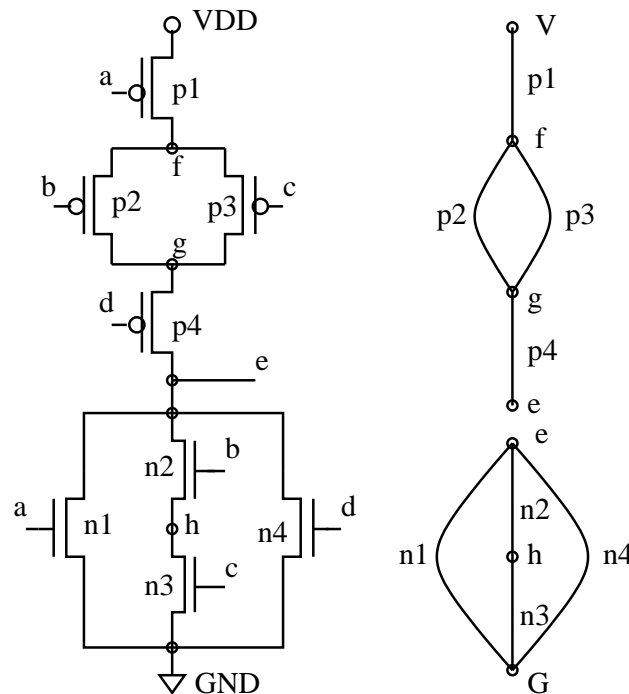


Figure 6.6: A schematic and layout graphs of a combinational cell

Calculations of the node degrees give:

	p-graph			n-graph			
node:	V	f	g	e	e	h	G
node degree:	1	3	3	1	3	2	3

The n-graph is Eulerian, because exactly two nodes are of odd degree, however, the p-graph is **not**, because more than two nodes are of odd degree, therefore, the p-diffusion trail without discontinuities does not exist.

The layout matrix resulting from the above graphs is of the form:

$$\mathcal{L}_1 = \begin{bmatrix} V & (p1) & f & (p2) & g & (p3) & f!e & (p4) & g \\ & a & & b & & c & & & d \\ G & (n1) & e & (n2) & h & (n3) & G & (n4) & e \end{bmatrix} \quad (6.4)$$

and the related layout is presented in Figure 6.7.

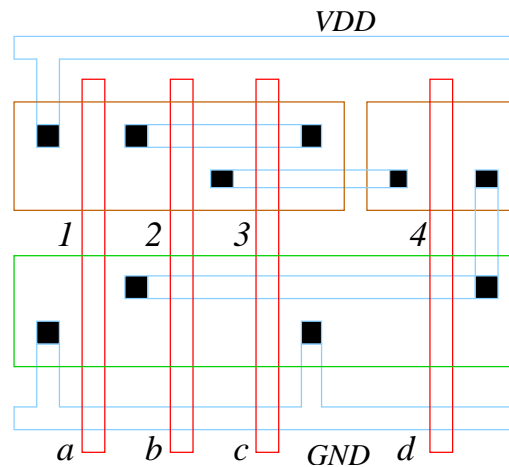


Figure 6.7: Layout of the cell with one discontinuity

There is one discontinuity in the p-diffusion. It can be removed by the following circuit manipulation.

By swapping the transistor $p4$ with the parallel connection of $p2$ and $p3$, which is equivalent to rewriting the logic equation for the p-MOS circuit in the form:

$$e_p = \bar{a} \cdot \bar{d} \cdot (\bar{b} + \bar{c})$$

we obtain the Eulerian layout graphs, which are presented in Figure 6.8. After this modifications both graphs are Eulerian, however, in both

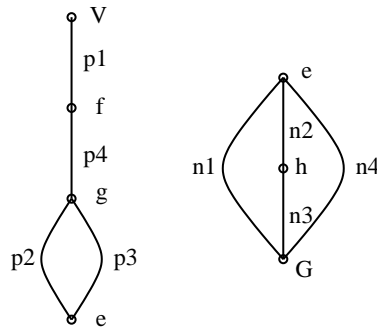


Figure 6.8: The improved layout graphs of the combinational cell

cases, there are two nodes of an odd degree (V, g and G, e), therefore, the diffusion paths must start at one of those nodes.

The layout matrix which satisfies all the requirements is of the following form:

$$\mathcal{L}_2 = \begin{bmatrix} V & (p1) & f & (p4) & g & (p3) & e & (p2) & g \\ & a & & d & & b & & c & \\ G & (n1) & e & (n4) & G & (n3) & h & (n2) & e \end{bmatrix} \quad (6.5)$$

The simplified layout of the above cell is given in Figure 6.9. Note

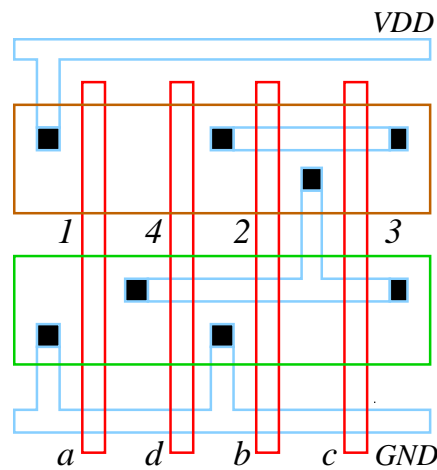


Figure 6.9: The improved layout of the combinational cell

that the reduction of the cell area is substantial.

6.2 1-bit Adders

A 1-bit adder is a fundamental building block of all arithmetic circuits. It has three inputs, a , b , c , and two outputs, d , s , known as the output carry and the sum, respectively. The 1-bit adder counts the number of ones at its three inputs and represents the result as a two-bit binary number. Hence, the defining arithmetic relationship between inputs and outputs can be written as:

$$a + b + c = (d, a)_2 = 2 \cdot d + s \quad (6.6)$$

All three inputs are equivalent, but normally c is called the input carry. Eqn. (6.6) can be converted into a truth table which describes the relationship between three adder inputs c , b , a and two adder outputs, d , s .

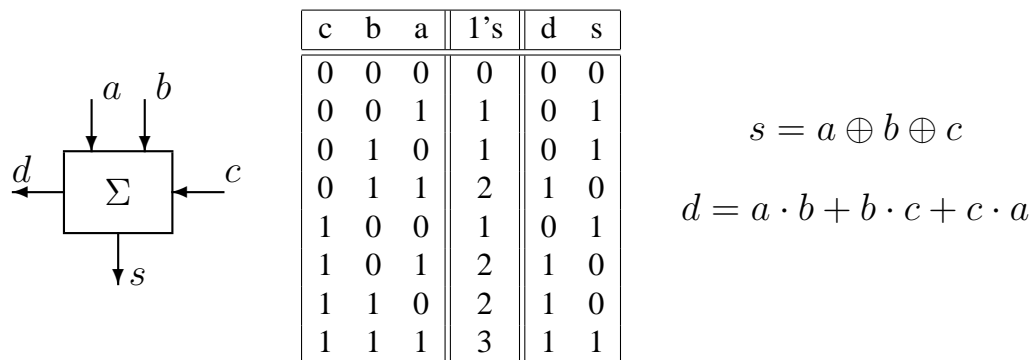


Figure 6.10: A 1-bit adder, its truth table and generic logic equations

From the truth table we can derive a generic logic equations for s and d which are shown in Figure 6.10.

We consider first three switch logic implementations of a 1-bit adder based on three XOR cells considered in sec. 5.9. Finally, we present a gate logic implementation with an interesting bridge-like structure of the sum circuit.

All three switch logic implementations are based on the following modified logic equations:

$$f = a \oplus b, \quad s = f \oplus c, \quad d = b \cdot \bar{f} + c \cdot f \quad (6.7)$$

Note that the intermediate signal f is used in generation of both the sum signal, s , and the output carry, d . It can be verified that these equations satisfy the truth table of Figure 6.10. The logic diagram equivalent to eqns (6.7) is given in Figure 6.11.

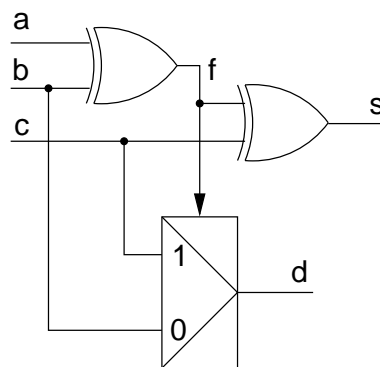


Figure 6.11: The logic diagram of a possible implementation of the 1-bit adder

6.2.1 The switch-logic, solution A

In our first implementation of a switch logic 1-bit adder we use the XOR cell presented in Figure 5.40 which is based on a buffered weak multiplexer. This implementation of the XOR function requires two complemented input signals, \bar{b} and \bar{c} . These complemented signals can be used in the carry out multiplexer to implement the complemented signal d .

$$\bar{d} = \bar{b} \cdot \bar{f} + \bar{c} \cdot f$$

We also need one buffering inverter in the carry out circuitry. The resulting 1-bit adder comprises **8 pairs** of transistors. A relevant schematic is shown in Figure 6.12.

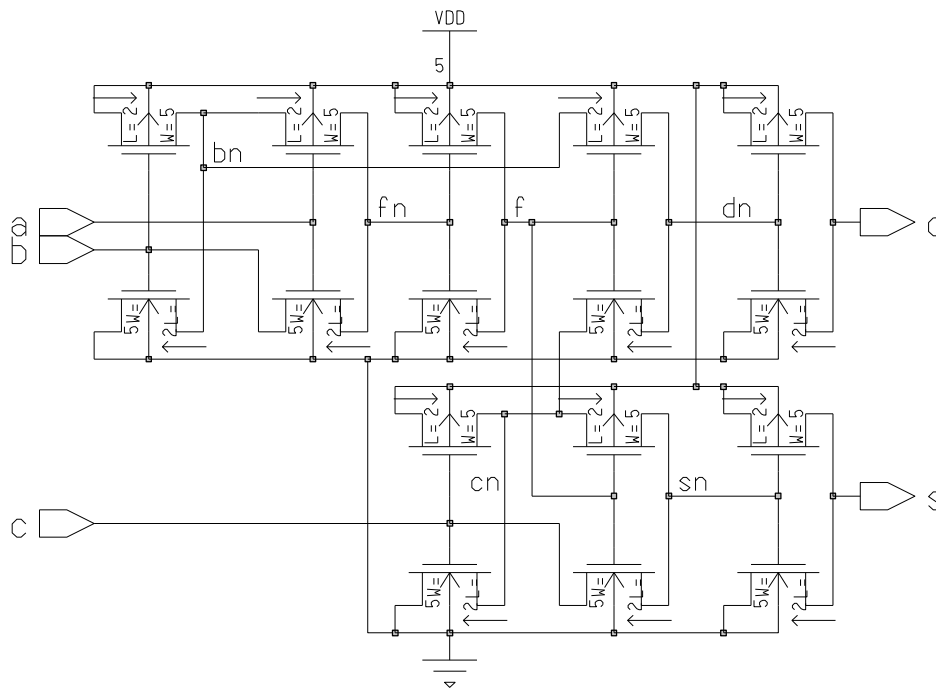


Figure 6.12: A schematic of a switch logic 1-bit adder — solution A

If we number the transistors in the top row of the schematic from

1 . . . 5 and those in the bottom row from 6 . . . 8, then the diffusion circuits (graphs) can be represented by the following table:

	p-MOS			n-MOS		
V	$(p1)$	bn		bn	$(n1)$	G
bn	$(p2)$	fn		fn	$(n2)$	b
V	$(p3)$	f		f	$(n3)$	G
bn	$(p4)$	dn		dn	$(n4)$	cn
V	$(p5)$	d		d	$(n5)$	G
V	$(p6)$	cn		cn	$(n6)$	G
cn	$(p7)$	sn		sn	$(n7)$	c
V	$(p8)$	s		s	$(n8)$	G

The orders of nodes are:

	V	G	b	bn	c	cn	fn	f	sn	s	dn	d	#odd
pMOS:	5			3		2	1	1	1	1	1	1	8
nMOS:		5	1	1	1	2	1	1	1	1	1	1	10

Since the number of odd degree nodes is high semi-Eulerian trails do not exist, and there will be a number of discontinuities both in p diffusion and in n diffusion. Optimization of the circuit by hand is difficult and one possible layout matrix can be written in the following form:

$$\begin{bmatrix}
 V (p1) & bn & (p2) & fn & bn & (p4) & dn & f & (p3) & V & (p5) & d & sn & (p7) & cn & (p6) & V & (p8) & s \\
 b & & a & & f & & & fn & & dn & & & & f & & c & & sn \\
 G (n1) & bn & b & (n2) & fn & dn & (n4) & cn & f & (n3) & G & (n5) & d & sn & (n7) & c & cn & (n6) & G & (p8) & s
 \end{bmatrix} \tag{6.8}$$

From the layout matrix (6.8) it is relatively straightforward to derive an equivalent stick diagram which is presented in Figure 6.13.

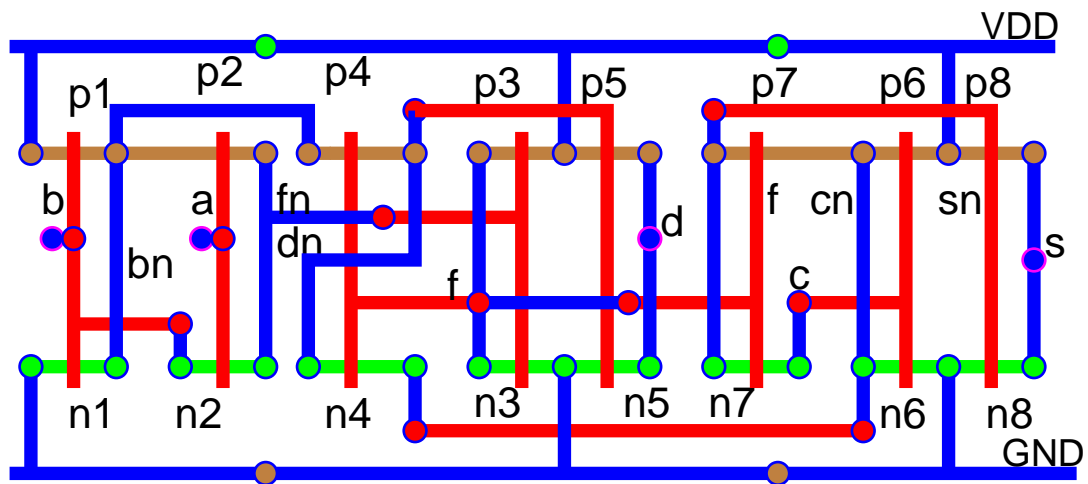


Figure 6.13: A stick diagram of the switch logic 1-bit adder, solution A

Simulation results for the switch logic 1-bit adder, solution A, implemented in the TSMC035 technology are presented in Figure 6.14. Propagation delays vary from 0.1ns to 0.6ns for the worst case when signal A goes low with B=0 and C=1 as indicated by the third cursor. Analyse the waveforms and the schematic to explain why this is the worst case.

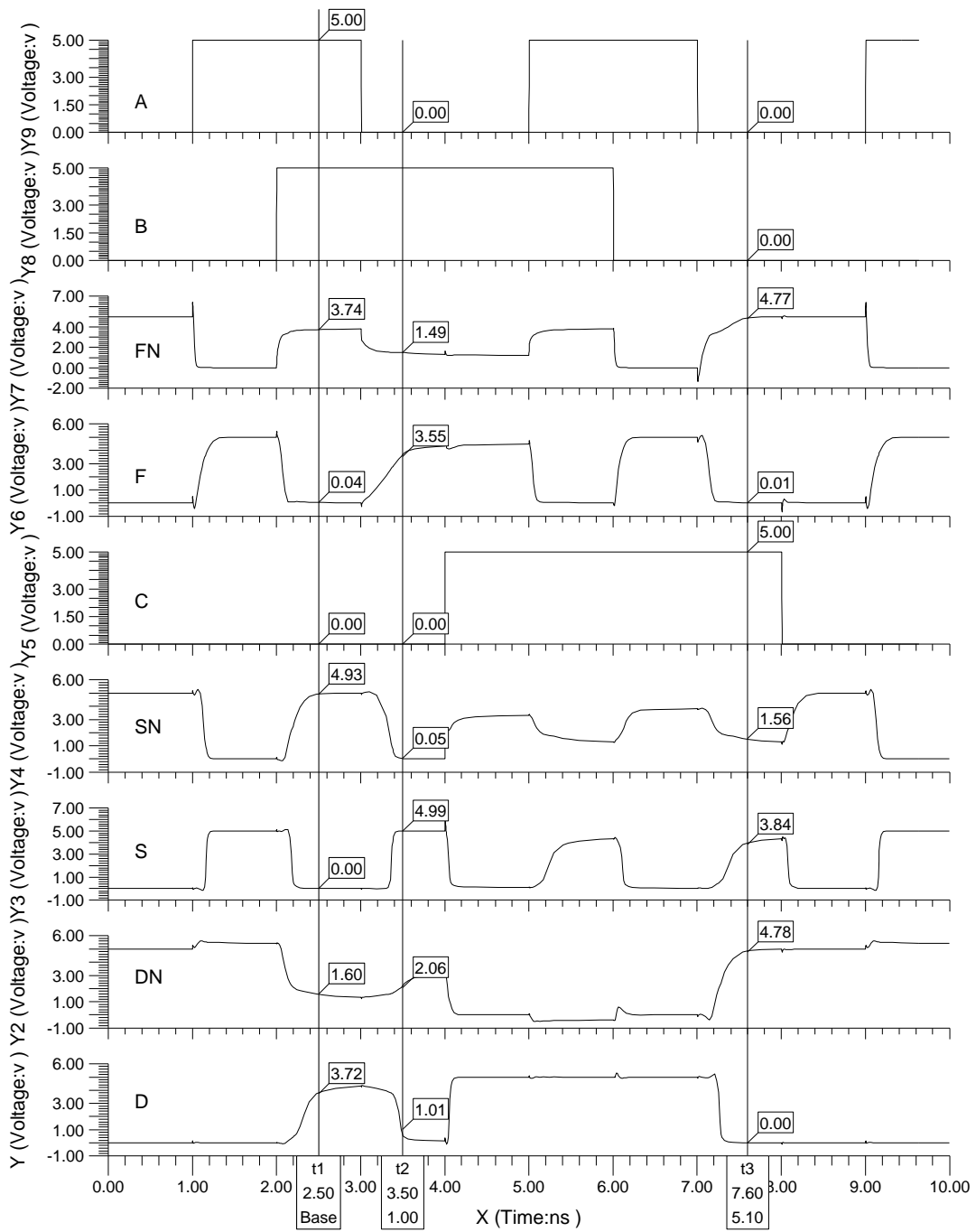


Figure 6.14: Simulation results for the switch logic 1-bit adder, solution A

6.2.2 The switch-logic, solution B

In this implementation we use the “best” XOR cells presented in Figure 5.43. The carry out circuit will be implemented using a weak multiplexer followed by two inverters:

$$d = a \cdot \bar{f} + c \cdot f$$

We now need **9 pairs** of transistors to build such an adder. A relevant schematic is shown in Figure 6.15.

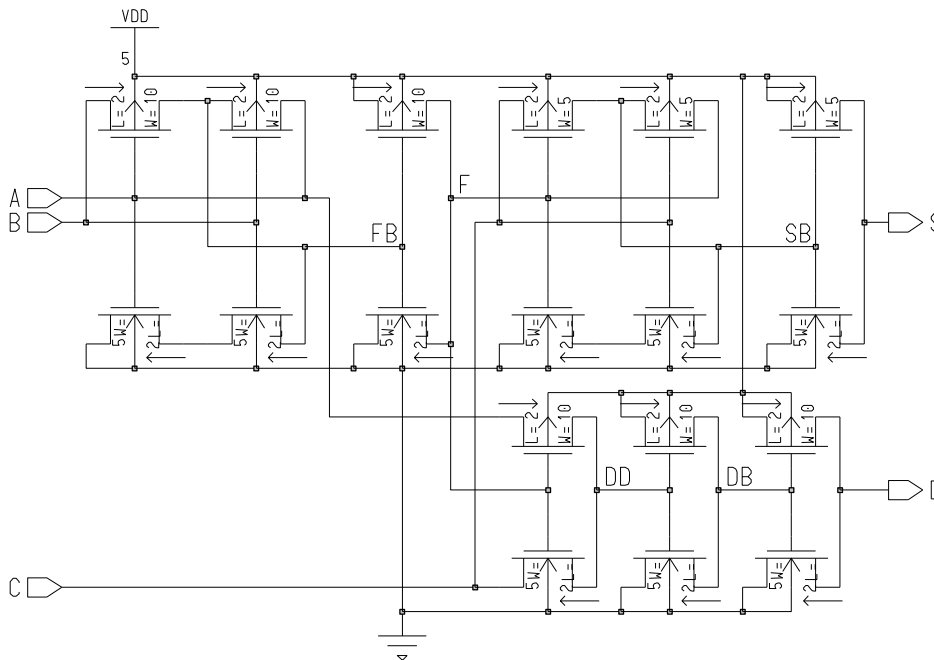


Figure 6.15: A schematic of a switch logic 1-bit adder, solution B

If we number the transistors in XOR circuits from 1 . . . 6 (the top row) and those used in the output carry multiplexer and inverters 7, 8, 9 (the bottom row), respectively, then the diffusion circuits (graphs) can be represented by the following table:

	p-MOS			n-MOS		
<i>a</i>	(<i>p1</i>)	<i>fb</i>	<i>n12</i>	(<i>n1</i>)	<i>G</i>	
<i>fb</i>	(<i>p2</i>)	<i>a</i>	<i>fb</i>	(<i>n2</i>)	<i>n12</i>	
<i>V</i>	(<i>p3</i>)	<i>f</i>	<i>f</i>	(<i>n3</i>)	<i>G</i>	
<i>sb</i>	(<i>p4</i>)	<i>c</i>	<i>n45</i>	(<i>n4</i>)	<i>G</i>	
<i>f</i>	(<i>p5</i>)	<i>sb</i>	<i>sb</i>	(<i>n5</i>)	<i>n45</i>	
<i>V</i>	(<i>p6</i>)	<i>s</i>	<i>s</i>	(<i>n6</i>)	<i>G</i>	
<i>a</i>	(<i>p7</i>)	<i>dd</i>	<i>dd</i>	(<i>n7</i>)	<i>c</i>	
<i>V</i>	(<i>p8</i>)	<i>db</i>	<i>db</i>	(<i>n8</i>)	<i>G</i>	
<i>V</i>	(<i>p9</i>)	<i>d</i>	<i>d</i>	(<i>n9</i>)	<i>G</i>	

where *n12*, *n45* indicates nodes between serially connected nMOS transistors. The orders of nodes are:

	<i>V</i>	<i>G</i>	<i>a</i>	<i>c</i>	<i>fb</i>	<i>f</i>	<i>sb</i>	<i>s</i>	<i>dd</i>	<i>db</i>	<i>d</i>	#odd
pMOS:	4		3	1	2	1	2	1	1	1	1	7
nMOS:		6		1	1	1	1	1	1	1	1	8

As in the previous solution the number of odd degree nodes is high, hence, semi-Eulerian trails do not exist, and there will be a number of discontinuities both in p diffusion and in n diffusion. Optimization of the circuit by hand is difficult and one possible layout matrix can be written in the following form:

$$\begin{bmatrix}
 s(p6) & V(p3) & f(p5) & sb(p4) & c! & b(p1) & fb(p2) & a(p7) & dd! & db(p8) & V(p9) & d \\
 sb & fb & c & f & a & b & f & dd & db \\
 s(n6) & G(n3) & f! & sb(n5) & (n4) & G(n1) & (n2) & fb! & c(n7) & dd! & db(p8) & G(p9) & d
 \end{bmatrix}
 \tag{6.9}$$

In Figure 6.16 we give the stick diagram equivalent to the layout matrix (6.9).

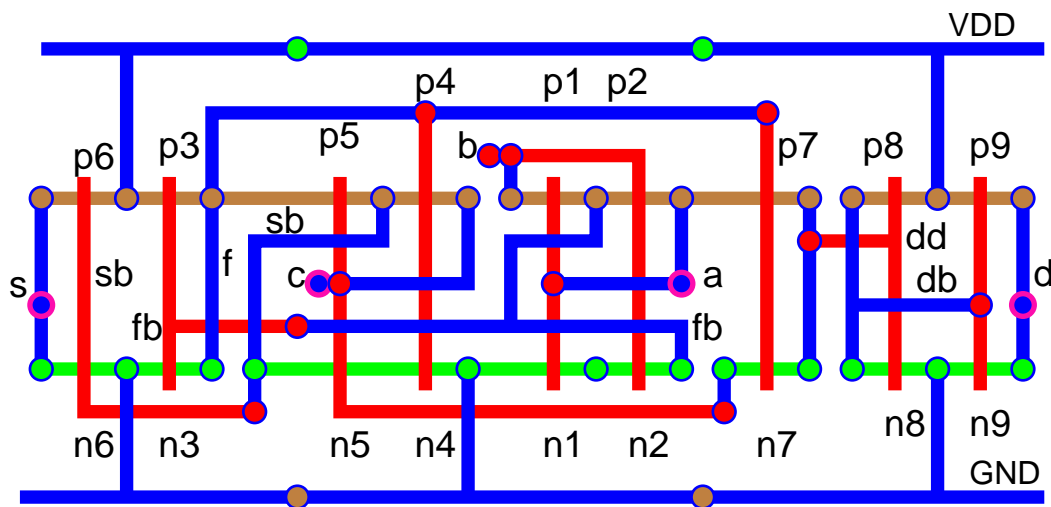


Figure 6.16: A stick diagram of the switch logic 1-bit adder, solution B

Simulation results in the TSMC035 technology are presented in Figure 6.17. Propagation delays vary from 0.2ns to 0.9ns for the worst case when signal A goes low with B=0 and C=1 as indicated by the third cursor. Analyse the waveforms and the schematic to explain why this is the worst case.

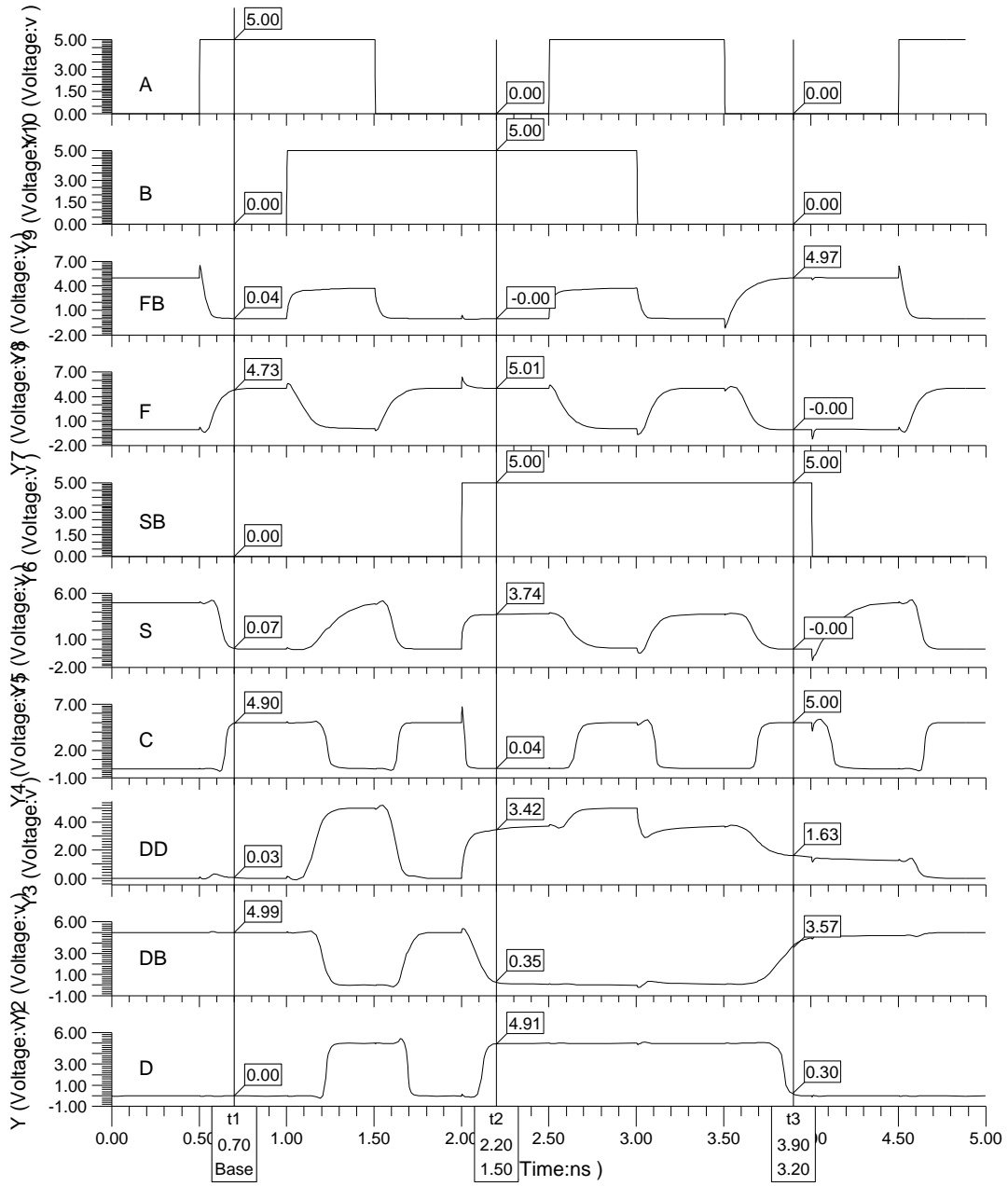


Figure 6.17: Simulation results of the switch logic 1-bit adder, solution B

6.2.3 The switch-logic: solution C

Finally, we consider a switch logic 1-bit adder based on the XOR cell built from a weak multiplexer and a transmission gate as in Figure 5.41. In addition, we implement the carry out multiplexer using transmission gates. The schematic of such an adder is shown in Figure 6.18. There are 11 pair of transistors, which makes the adder structure

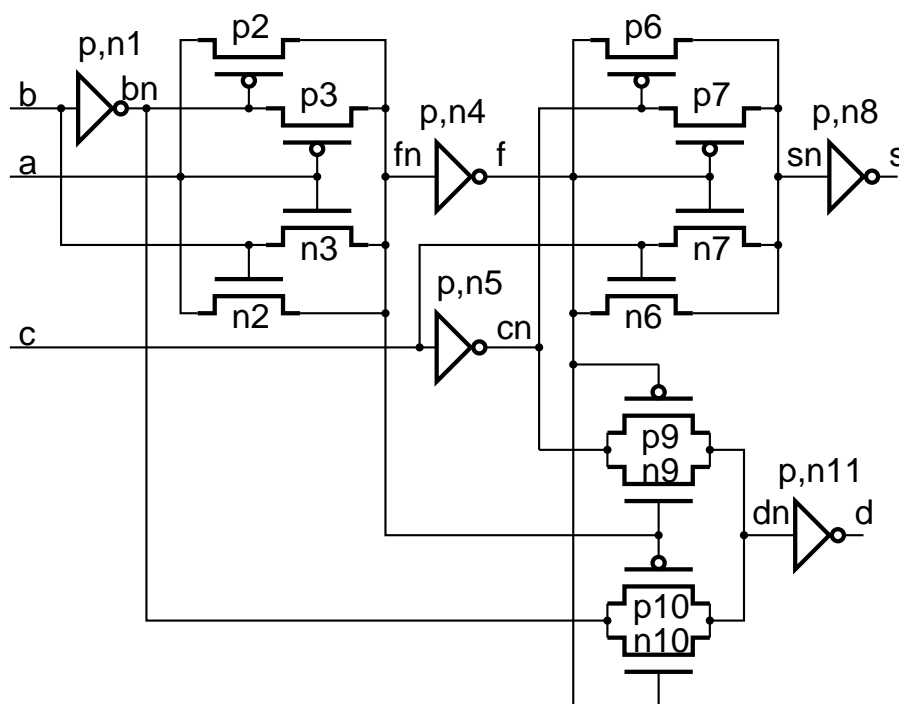


Figure 6.18: Symbolic schematic of a switch logic 1-bit adder based on transmission gates, solution C.

rather complicated, also because the transistors in transmission gates are driven by different signals.

The first step in designing the circuit layout is to for a list of transistors and electric nodes, which can be of the following form:

p-MOS			Gate signals	n-MOS		
<i>V</i>	(<i>p1</i>)	<i>bn</i>	<i>b</i>	<i>bn</i>	(<i>n1</i>)	<i>G</i>
<i>a</i>	(<i>p2</i>)	<i>fn</i>	<i>bn b</i>	<i>fn</i>	(<i>n2</i>)	<i>a</i>
<i>bn</i>	(<i>p3</i>)	<i>fn</i>	<i>a</i>	<i>fn</i>	(<i>n3</i>)	<i>b</i>
<i>V</i>	(<i>p4</i>)	<i>f</i>	<i>fn</i>	<i>f</i>	(<i>n4</i>)	<i>G</i>
<i>V</i>	(<i>p5</i>)	<i>cn</i>	<i>c</i>	<i>cn</i>	(<i>n5</i>)	<i>G</i>
<i>f</i>	(<i>p6</i>)	<i>sn</i>	<i>cn c</i>	<i>sn</i>	(<i>n6</i>)	<i>f</i>
<i>cn</i>	(<i>p7</i>)	<i>sn</i>	<i>f</i>	<i>sn</i>	(<i>n7</i>)	<i>c</i>
<i>V</i>	(<i>p8</i>)	<i>s</i>	<i>sn</i>	<i>s</i>	(<i>n8</i>)	<i>G</i>
<i>cn</i>	(<i>p9</i>)	<i>dn</i>	<i>f fn</i>	<i>dn</i>	(<i>n9</i>)	<i>cn</i>
<i>bn</i>	(<i>p10</i>)	<i>dn</i>	<i>fn f</i>	<i>dn</i>	(<i>n10</i>)	<i>bn</i>
<i>V</i>	(<i>p11</i>)	<i>d</i>	<i>dn</i>	<i>d</i>	(<i>n11</i>)	<i>G</i>

The orders of nodes (diffusion only) are:

	<i>V</i>	<i>G</i>	<i>a</i>	<i>b</i>	<i>bn</i>	<i>c</i>	<i>cn</i>	<i>fn</i>	<i>f</i>	<i>sn</i>	<i>dn</i>	<i>s</i>	<i>d</i>	#odd
pMOS:	5		1		3		3	2	2	2	2	1	1	6
nMOS:		5	1	1	2	1	2	2	2	2	2	1	1	6

Design of the layout provides a challenge, as the number of the odd degree nodes is significant. We can start with drawing graphs of the diffusion circuits and then partition the circuits into sub-circuits for which semi-Eularian trails exist. One possible solution is described by the following layout matrix:

$$\begin{bmatrix}
 a(p2)fn(p3)bn!V(p1)bn(p10)dn(p9)cn(p5)V(p4)f(p6)sn(p7)cn!s(p8)V(p11)d \\
 bn \quad a \quad b \quad fn \quad f \quad c \quad fn \quad cn \quad f \quad sn \quad dn \\
 b \quad a \quad b \quad f \quad fn \quad c \quad fn \quad c \quad f \quad sn \quad dn \\
 a(n2)fn(n3)b!G(n1)bn(n10)dn(n9)cn(n5)G(n4)f(n6)sn(n7)c!s(n8)G(n11)d
 \end{bmatrix}
 \tag{6.10}$$

6.2.4 A Cell with a bridge-like structure

All circuit that we have considered by now consists of serial/parallel connections of transistors. In this example we consider a circuit having a different topology, namely, a bridge-like structure. A schematic of a 1-bit adder with such a structure is shown in Figure 6.19. The bridge implements the sum using a complement of the output carry. The output carry is implemented using a composite gate concept. Implementation equations for this adder are of the form:

$$\begin{aligned} \text{Output Carry, } d &= a \cdot b + (a + b) \cdot c \\ \text{p-MOS: } d_p &= \bar{d} = \bar{a} \cdot \bar{b} + (\bar{a} + \bar{b}) \cdot \bar{c} \\ \text{n-MOS: } d_n &= d = a \cdot b + (a + b) \cdot c \\ \text{Sum, } s &= a \oplus b \oplus c \\ \text{p-MOS: } s_p &= \bar{s} = (\bar{a} + \bar{b} + \bar{c}) \cdot d + \bar{a} \cdot \bar{b} \cdot \bar{c} \\ \text{n-MOS: } s_n &= s = (a + b + c) \cdot \bar{d} + a \cdot b \cdot c \end{aligned}$$

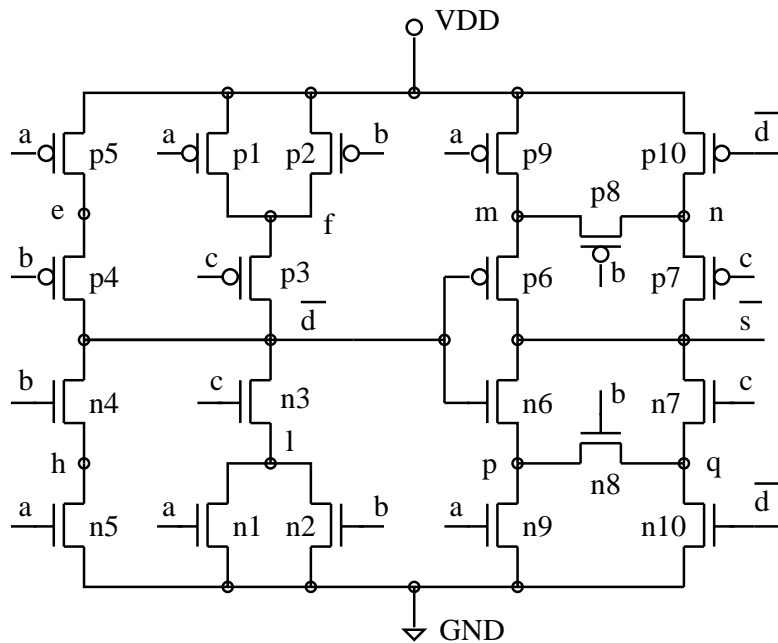


Figure 6.19: A 1-bit adder with the bridge-like sum circuit

The structure of the adder is presented in Figure 6.20 in the form of the graphs for the p-MOS and n-MOS parts of the circuit.

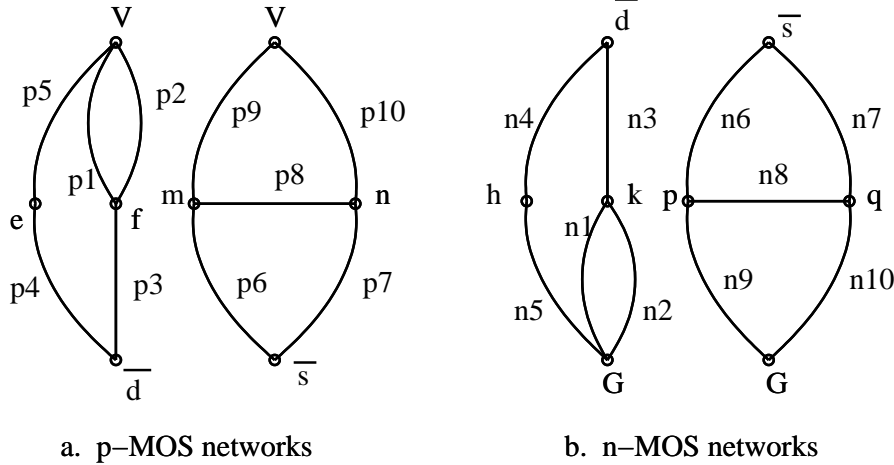


Figure 6.20: Circuit graphs of the adder

There are two interesting points to be made about this circuit. Firstly, it can be noted that the structures of the p-MOS part and the n-MOS part of the output carry circuit are identical, despite complementation, because the output carry is described by the symmetric boolean function. Secondly, both parts of the sum circuit have an interesting ‘bridge-like’ structure which is not of a serial/parallel form. Although this fact makes formal properties of the underlying graphs to be more complicated, it does not change the design procedure. As usual, the next step, after obtaining the p-MOS and n-MOS graphs, is to calculate the degree of each node. The results of this calculations are presented in the following table:

	p-graph (\bar{d})				p-graph (\bar{s})			
node:	V	e	\bar{d}	f	V	m	\bar{s}	n
node degree :	3	2	2	3	2	3	2	3
	n-graph (\bar{d})				n-graph (\bar{s})			
node:	\bar{d}	h	G	k	\bar{s}	p	G	q
node degree :	2	2	3	3	2	3	2	3

All of the graphs meet the conditions to be Eulerian. In order to obtain an Eulerian trail, we have to traverse the graphs at nodes with odd degrees, which results in the following layout matrices:

$$\mathcal{L}_D = \begin{bmatrix} V & (p5) & e & (p4) & \bar{d} & (p3) & f & (p2) & V & (p1) & f \\ & a & & b & & c & & b & & a & \\ G & (n5) & h & (n4) & \bar{d} & (n3) & k & (n2) & G & (n1) & k \end{bmatrix} \quad (6.11)$$

$$\mathcal{L}_S = \begin{bmatrix} n & (p10) & V & (p9) & m & (p8) & n & (p7) & \bar{s} & m \\ & \bar{d} & & a & & b & & c & & \bar{d} \\ q & (n10) & G & (n9) & p & (n8) & q & (n7) & \bar{s} & (n6) & p \end{bmatrix} \quad (6.12)$$

From the layout matrices (6.11) and (6.12) we note that although both circuits are Eulerian, they cannot be laid together without a gap, because the bordering nodes are different.

If, in addition, we add inverters for \bar{d} and \bar{s} signals, the stick diagram of the 1-bit adder can be arranged as presented in Figure 6.21.

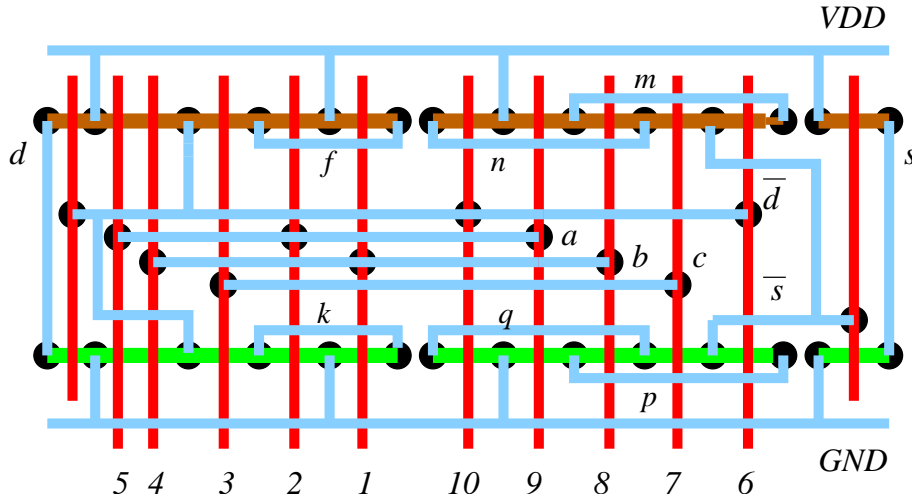


Figure 6.21: A stick diagram of the 1-bit adder

The stick diagram, which symbolically specifies the actual topology in silicon, can be easily converted into the mask layout.

6.3 Building Blocks of Sequential Circuits

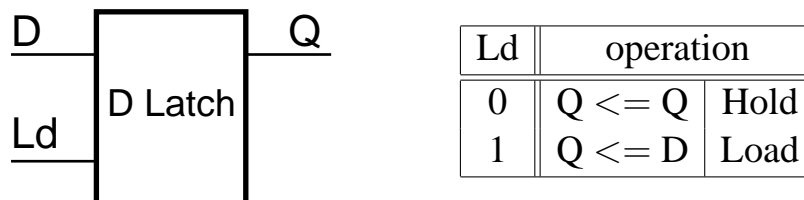
Basic blocks of sequential circuits can be divided into two groups:

Latches are **level-triggered asynchronous** sequential circuits containing just one feedback which is equivalent to two asynchronous states.

Flip-Flops are **edge-triggered** sequential circuits which contain two synchronous states, or at least four asynchronous states that is at least two internal feedback loops.

6.3.1 D Latch

The D latch is an asynchronous sequential circuit specified by the following **function table**:



The function table can be converted into an equivalent **state equation**:

$$Q^+ = \overline{Ld} \cdot Q + Ld \cdot D \quad (6.13)$$

where Q represents the **current** value of the **state** signal and Q^+ represents its **next** value. Equation (6.13) give rise to the generic implementation of the D latch as in Figure 6.22 in which a multiplexer sends to the output either the current state Q or the input signal D . The feedback loop connects the “next state” with the current one.

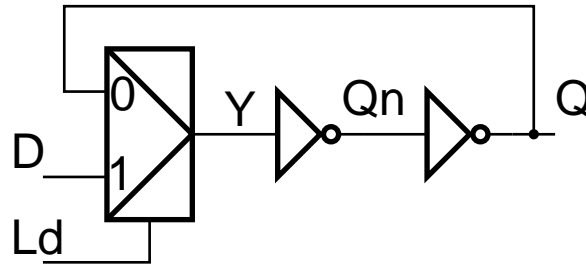


Figure 6.22: A generic implementation of the D Latch.

A weak multiplexer implementation

If a “weak” multiplexer is employed, then the transistor level schematic equivalent to the generic circuit from Figure 6.22 can be of the form as in Figure 6.23.

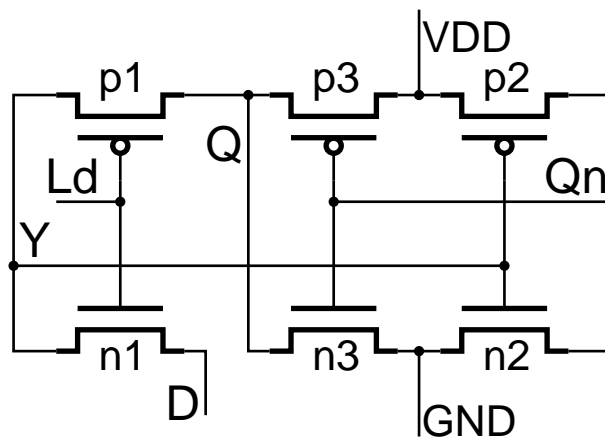


Figure 6.23: A transistor level schematic of the D Latch based on the weak multiplexer.

The layout matrix equivalent to the schematic of Figure 6.23 can be written as follows:

$$\mathcal{L} = \begin{bmatrix} Y & (p1) & Q & (p3) & V & (p2) & Qn \\ & Ld & & Qn & & Y & \\ D & (n1) & Y ! Q & (n3) & G & (n2) & Qn \end{bmatrix}$$

Note that in the layout matrix we have flipped the n1 nMOS transistor to shorten the net 'Y'. The resulting stick diagram that flows from the schematic and the layout matrix is shown in Figure 6.24.

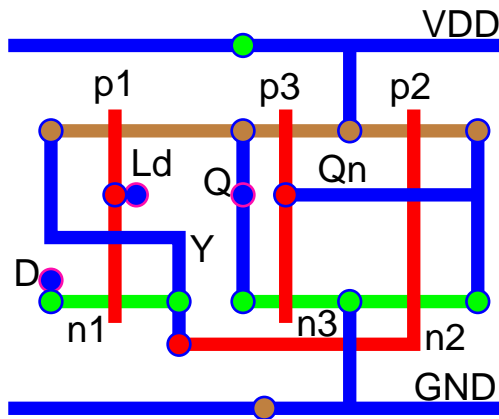


Figure 6.24: The stick diagram of a D latch based on a weak multiplexer.

Note that the input-output ports are available on the metal2 layer and are ready to be connected using the vertical metal2 paths. The stick diagram has been converted into a circuit layout in the TSMC 0.35μ technology which is presented in Figure 6.25.

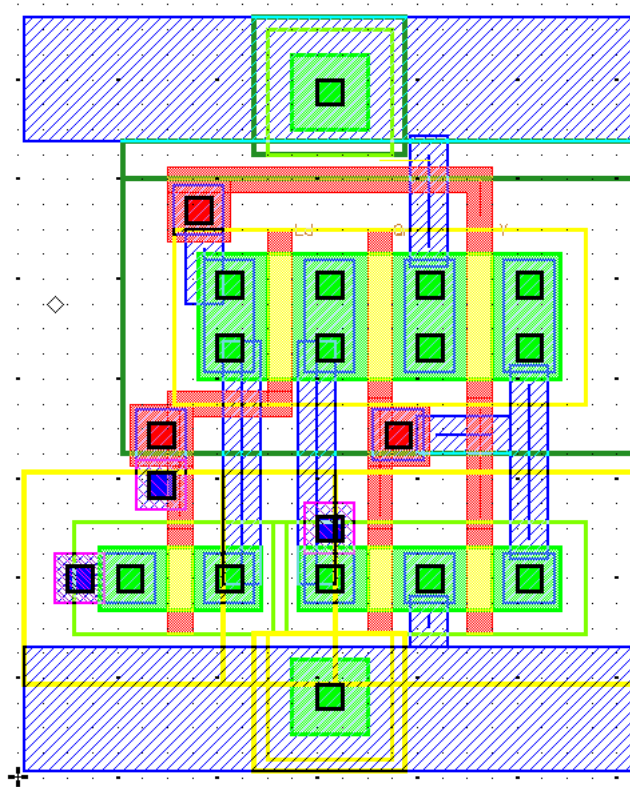


Figure 6.25: A possible layout of a D latch in the TSMC 0.35μ technology.

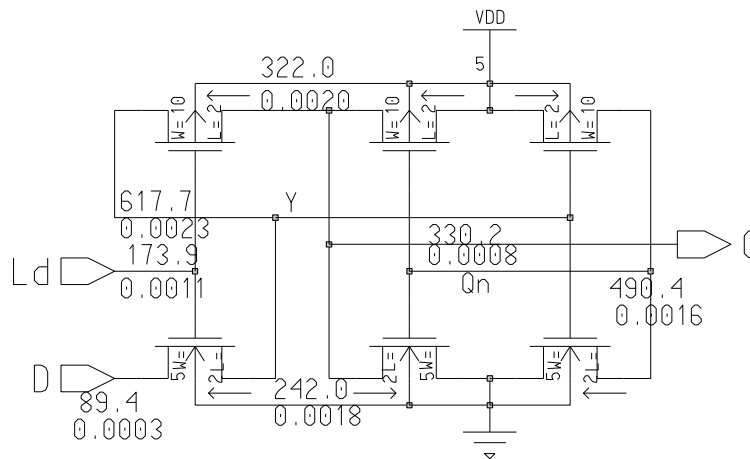


Figure 6.26: A back annotated schematic

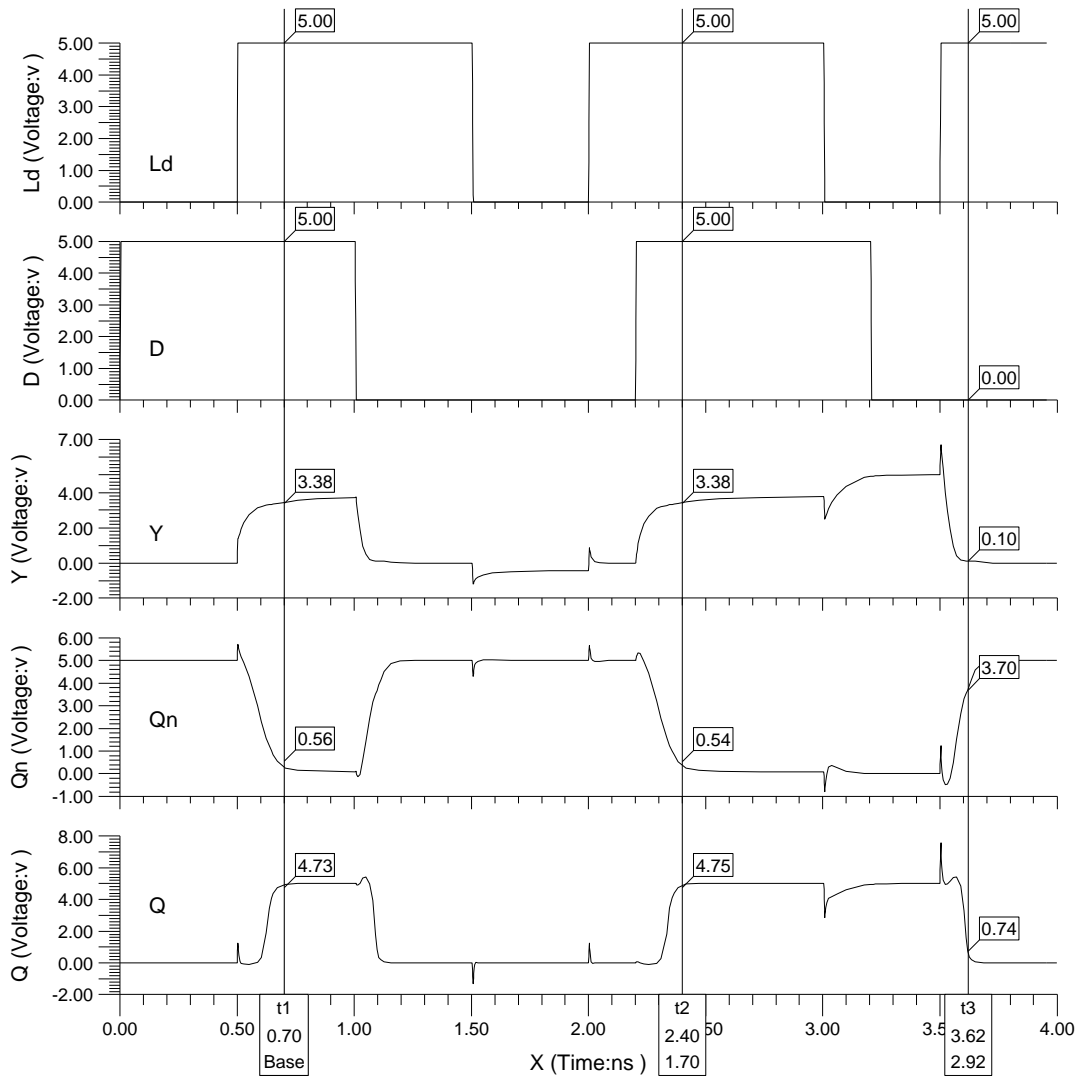


Figure 6.27: Simulation results

A composite gate implementation

From eqn (6.13) we can also derive a composite gate implementation of the D latch. This equation can be written in the following “inverted” form:

$$Qn = \overline{A + D \cdot Ld} , \quad A = \overline{Q + Ld} , \quad Q = \overline{Qn}$$

which is equivalent to the logic diagram as in Figure 6.25.

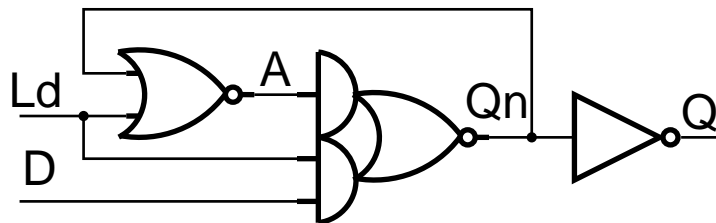


Figure 6.28: A logic diagram of the D latch based on a composite gate

The logic diagram of Figure 6.28 can be converted into a transistor-level schematics as presented in Figure 6.29. In the

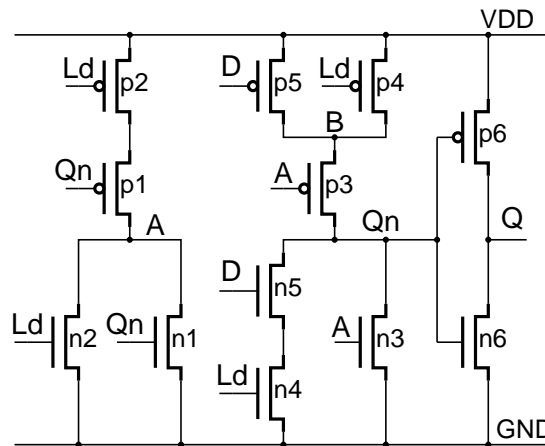


Figure 6.29: A schematic of the D latch based on a composite gate

schematics you should identify the NOR gate, the AOI gate and the inverter.

In a process of a systematic design the next step is to convert the schematic into two circuits graphs, for the nMOS and pMOS sections, respectively, as presented in Figure 6.30.

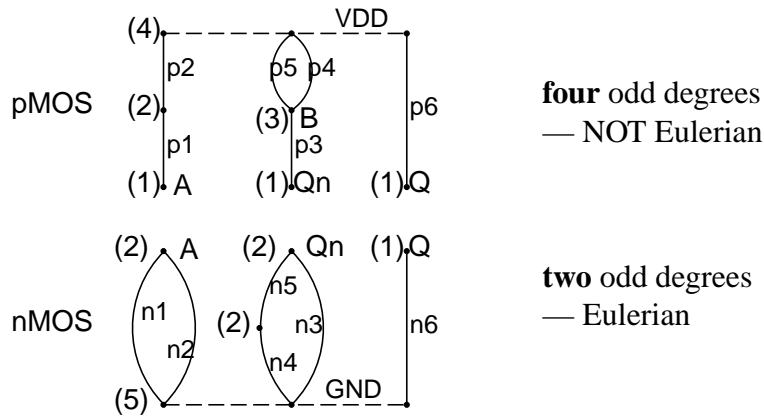


Figure 6.30: Circuit graphs of the D latch based on a composite gate

The pMOS is not Eulerian and as a result of the p diffusion area cannot be continuous. In order to minimise the number of gaps the p-graph has been rearranged as in Figure 6.31.

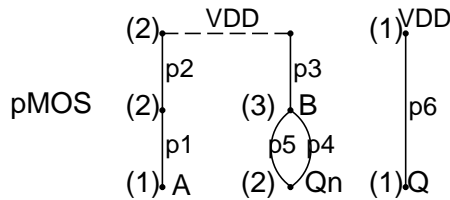


Figure 6.31: The graph of the p-section after rearrangements

The layout matrix which results from the above graphs can be written in the following form:

$$\mathcal{L} = \begin{bmatrix} A (p1) & (p2) V & (p3) B & p4 & Qn & (p5) B & ! V & (p6) Q \\ Ld & Qn & A & Ld & D & Qn & & \\ G (n1) & A (n2) & G (n3) & Qn (n4) & (n5) & G & (n6) & Q \end{bmatrix}$$

The stick diagram which results from the above layout matrix is presented in Figure 6.32.

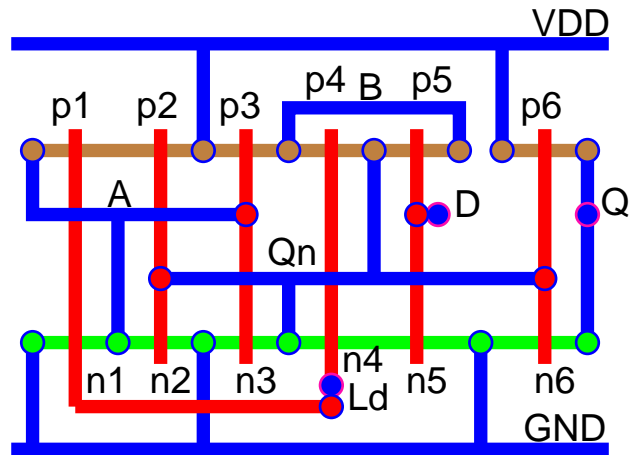


Figure 6.32: A Stick diagram of the D latch based on a composite gate.

6.3.2 Designing an Edge-Triggered D-type Flip-Flop

An edge-triggered D-type flip-flop is an example of a relatively complex **asynchronous** sequential circuit. Once designed it will be a building block practically all registers and counters.

There are formal methods of designing asynchronous sequential circuits, however, because they are relatively complex, we start here with the following **state diagram** which precisely specifies the behaviour of the flip-flop.

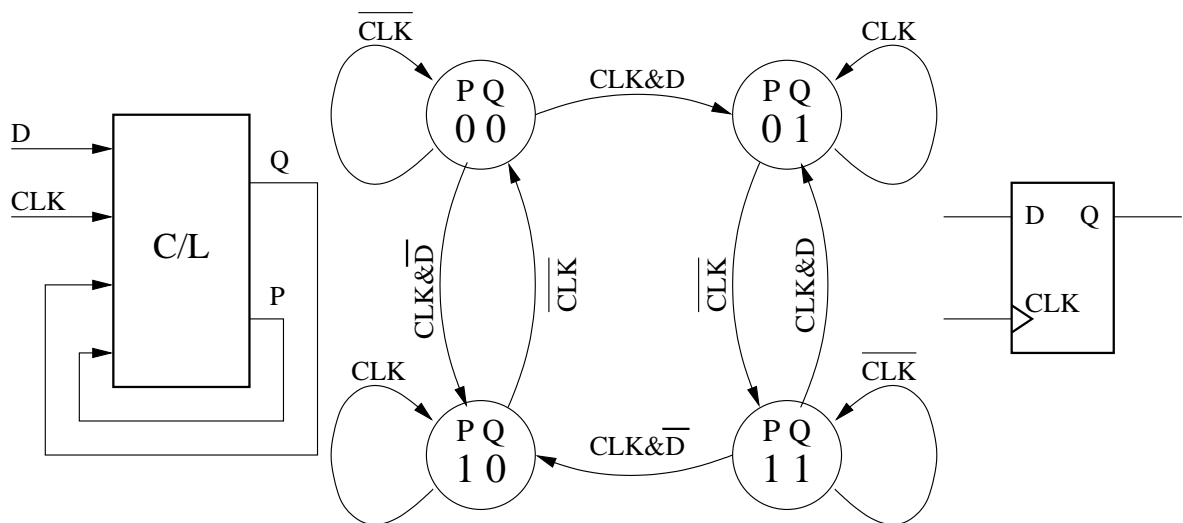


Figure 6.33: A block-diagram, the **state diagram** and the symbol of a positive-edge-triggered D-type flip-flop

There are **four states** coded by 2-bit binary words which represent **two state signals**, P and Q. There are two input signals, D (data) and CLK (clock). Both state signals can be used as outputs, the Q signal being of the principal interest.

The expected behaviour described by the state diagram is that during the **rising edge** of the clock signal the 1-bit data from the input D is loaded into the flip-flop, that is,

$$Q \leq D$$

Outside the rising edge of the clock signal the state Q is to be unchanged, regardless of the variations of the D input and possibly the second state signal, P.

Details of such a behaviour should be visible by inspecting the state diagram:

- Assume that PQ=00. If CLK=0, we remain in this stage. If CLK=1, we go either to state PQ=01, or PQ=10 depending on the value of the D signal.
- If CLK=1 and we have just move to PQ=01 or 10, we stay in such a state regardless of the value of the signal D. In other words, the value of signal D is sampled only when we move from PQ=00 (11) to PQ=01 or 10. This is the essence of **sensitivity to the positive edge** of the clock signal.
- In states PQ=01 or 10 we wait for clock signal to go to zero and then we go either to PQ=11 or 00, respectively.

The above behaviour can be best described by the following timing diagrams:

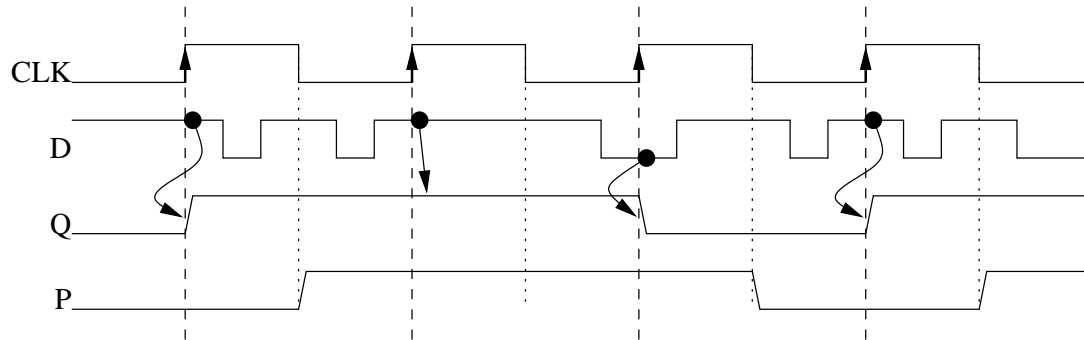


Figure 6.34: The timing diagram of the edge-triggered D-type flip-flop

In order to design the flip-flop we have to design a combinatorial circuit (C/L in Figure 6.33) which implements the required behaviour specified by the state diagram.

For this, we convert the **state diagram** into the **state transition** table. The state transition table describes the relationship between the current state signal, P, Q, the input signals, CLK, D, and the next state signals P^+ , Q^+ . The symbol * denotes the don't care conditions.

CLK	D	P	Q	P^+	Q^+
0	*	0	0	0	0
1	0	0	0	1	0
1	1	0	0	0	1
0	*	0	1	1	1
1	*	0	1	0	1
0	*	1	0	0	0
1	*	1	0	1	0
0	*	1	1	1	1
1	0	1	1	1	0
1	1	1	1	0	1

The state transition table is equivalent to the truth table for the following combinatorial circuits:

$$P^+ = f_P(\text{CLK}, D, P, Q)$$

$$Q^+ = f_Q(\text{CLK}, D, P, Q)$$

The functions f_P and f_Q can be obtained from the following Karnaugh maps generated from the state transition table.

		P^+						Q^+			
CLK	D	00	01	11	10	CLK	D	00	01	11	10
PQ						PQ					
00	00	0	0	0	1	00	00	0	0	1	0
01	01	1	1	0	0	01	01	1	1	1	1
11	11	1	1	0	1	11	11	0	0	0	0
10	10	0	0	1	1	10	10	0	0	1	1

From the Karnaugh maps it is possible to find a suitable gate implementation of the combinatorial part of the flip-flop. Two feedback loops, for P and Q, complete the design.

As an exercise please complete the design developing a logic diagram, a transistor level schematic, a stick diagram and a possible circuit layout.

Verify your design using your favourite CAD package.

It should be realised that the gate logic solution for the edge-triggered D flip-flop is unnecessary big and the smaller solutions are based on the switch logic. In Figure 6.35 we present a DFF from the AMI 0.5μ technology which is based on the **tri-state inverters**.

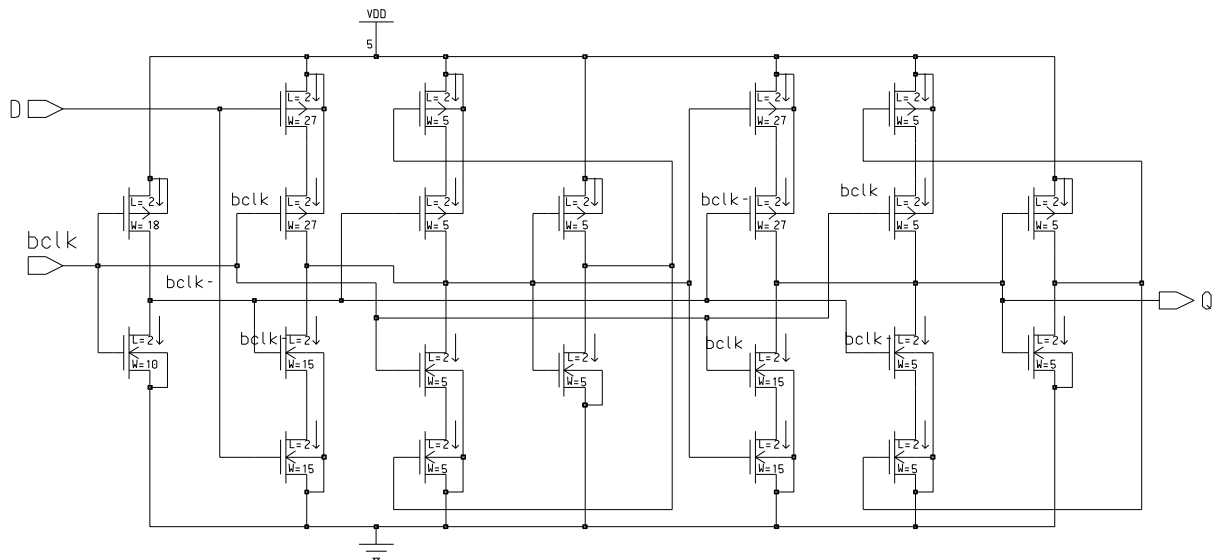


Figure 6.35: A schematic of a D flip-flop based on tri-state inverters (from the AMI 0.5μ library)

As an exercises, analyse this solution in detail and derive an equivalent logic/block diagram.

6.3.3 Edge-Triggered T Flip-Flop

The edge-triggered T flip-flop can be easily built from a D flip-flop so that on the rising edge of the clock, CLK, one of the following two operations are performed:

T	operation	
0	$Q \leq Q$	hold
1	$Q \leq \bar{Q}$	toggle

Alternatively, the T flip-flop can be designed as an asynchronous sequential circuit with the state diagram as in Figure 6.36.

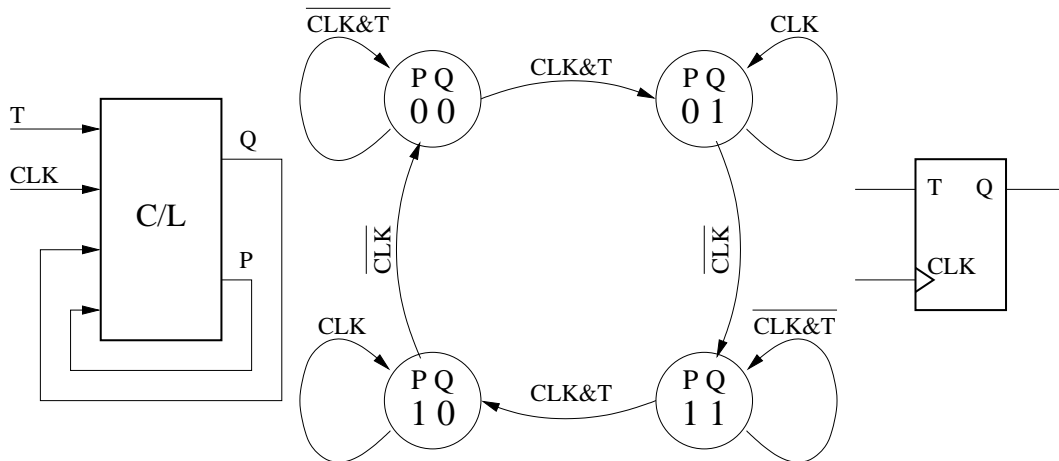


Figure 6.36: A block-diagram, the **state diagram** and the symbol of a positive-edge-triggered T-type flip-flop

Comparing with the D flip-flop the state diagram is a bit simpler which should result in a simpler implementation.

In order to design the flip-flop we convert the state diagram into the state transition table, possible in the form of a Karnaugh map. The resulting map is as follows:

		$P^+ \quad Q^+$			
Clk T	PQ	00	01	11	10
00	00	00	00	01	00
01	01	11	11	01	01
11	11	11	11	10	11
10	10	00	00	10	10

From the above table we can derive the following logic equations:

$$P^+ = Q \cdot \overline{Clk} + P \cdot Clk$$

$$Q^+ = Q \cdot \overline{Clk} + Clk \cdot (T \cdot \overline{P} + \overline{T} \cdot Q)$$

The signal P^+ can be generated using a 2-to-1 multiplexer driven by Clk which switches between Q and P .

To generate the signal Q^+ we can use two 2-to-1 multiplexers. The first one is driven by T which switches between Q and P . The second multiplexer is driven by Clk which switches between Q and the output from the first one.

A weak-multiplexer implementation of the positive edge triggered T flip-flop is given in Figure 6.37.

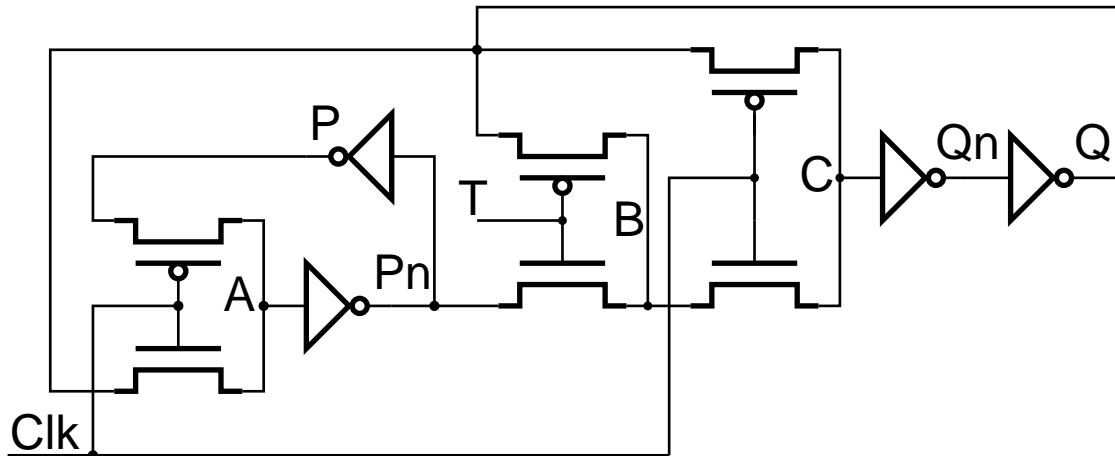


Figure 6.37: A schematic of a positive edge triggered T flip-flop based on weak multiplexers.

The above implementation consists of three weak 2-to-1 multiplexers generating signals A, B, C. Signals which are fed back are amplified by a pair of inverters. The complete implementation contains of seven pair of MOS transistors.

Exercise: Derive the layout matrix and the resulting circuit layout.